UNIVERSITY OF BIRMINGHAM
University of Birmingham Research at Birmingham

TIE algorithm: A layer over clustering-based taxonomy generation for handling an evolving data

Irfan, Rabia; Khan, Sharifullah; Rajpoot, Kashif; Qamar, Ali Mustafa

DOI:
10.1631/FITEE.1700517

License:
None: All rights reserved

Document Version
Peer reviewed version

Citation for published version (Harvard):
Irfan, R, Khan, S, Rajpoot, K & Qamar, AM 2018, 'TIE algorithm: A layer over clustering-based taxonomy generation for handling an evolving data', *Frontiers of Information Technology and Electronic Engineering*, vol. 19, no. 6, pp. 763–782. https://doi.org/10.1631/FITEE.1700517

Link to publication on Research at Birmingham portal

Publisher Rights Statement:
This is a post-peer-review, pre-copyedit version of an article published in Frontiers of Information Technology & Electronic Engineering. The final authenticated version is available online at: http://dx.doi.org/10.1631/FITEE.1700517

General rights
Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

•Users may freely distribute the URL that is used to identify this publication.
•Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
•User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
•Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

Take down policy
While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

Download date: 05. May. 2023

TIE algorithm: A layer over clustering-based taxonomy generation for handling an evolving data

Rabia Irfan[‡1], Sharifullah Khan[1], Kashif Rajpoot[1,2], Ali Mustafa Qamar[3]

([1]School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan)
([2]School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom)
([3]Department of Computer Science, College of Computer, Qassim University, Al Malida, Buraydah Saudi Arabia)
[‡]E-mail: 12phdrirfan@seecs.edu.pk

Abstract: Taxonomy is generated to effectively organize and access data that is large in volume, as taxonomy is a way of representing concepts that exist in data. It needs to be evolved to reflect changes occur continuously in data. Existing automatic taxonomy generation techniques do not handle the evolution of data, therefore their generated taxonomies do not truly represent the data. The evolution of data can be handled either by regenerating taxonomy from scratch, or incrementally evolving taxonomy whenever changes occur in the data. The former approach is not economical subject to time and resources. Taxonomy incremental evolution (TIE) algorithm, proposed in this paper, is a novel attempt to handle an evolving data. It serves as a layer over an existing clustering-based taxonomy generation technique and incrementally evolves an existing taxonomy. The algorithm was evaluated on scholarly articles selected from computing domain. It was found that the algorithm evolves taxonomy in a considerably shorter period of time, having better quality per unit time as compared to the taxonomy regenerated from scratch.

Key words: Taxonomy; Clustering algorithms; Information science; Knowledge management; Machine learning

1 Introduction

Data is produced in a large volume on daily basis nowadays (Turner, 2014). According to Computer Science Corporation (Koff and Gustafson, 2012) report on data revolution, experts are expecting 4300% increase in annual data generation by the year 2020. A major chunk of this data is unstructured text data, like scholarly articles, technical reports, organizational policy documents, etc. (Blumberg and Atre, 2003), which dominates 80% of the data industry these days (Koff and Gustafson, 2012). This exponential growth of data is so overwhelming that it can actually lead to the possibility of missing new directions and emerging ideas, rather than discovering novel insights from it. To utilize this data effectively, it should be processed and transformed into valuable information. The organization of this information in a structured form, like taxonomy, can be helpful in utilizing it timely, effectively and accurately. Taxonomy is a structured organization of hierarchical or parent-child relationships of concepts present in data (Paukkeri *et al.*, 2012). Muller *et al.* (1999) define taxonomy as a thematic structure inherent in data. There are many applications of taxonomy. It is an effective mean of categorizing and organizing data (Sujatha and Krishna Rao, 2011). It provides standardization, so that less interoperability issues may arise (Engel *et al.*, 2010). Furthermore, it serves as a foundation structure for content and knowledge management (Hedden, 2010), information search and navigation (Sanchez and Moreno, 2004), analytics and text mining (Li and Anand, 2009; Dawelbait *et al.*, 2010; Weng and Liu, 2004; Spangler *et al.*, 2006; Camina, 2010).

Currently, we find many automatic taxonomy generation techniques, which are producing taxonomy effectively for small to large data sets. However, the existing techniques ignore the fact that data is growing at an extremely rapid pace and changes occur in data should also be reflected on taxonomy. The taxonomy that does not consider changes occur in data, cannot truly represent the data. There can be two ways to handle changes occur in data. One way is to regenerate taxonomy from scratch, which is not economical subject to time and resources. Another way is to evolve taxonomy incrementally. Taxonomy incremental evolution (TIE) algorithm, proposed in this paper, is a novel attempt in this direction. The proposed algorithm serves as a layer over an existing clustering-based taxonomy generation technique and incrementally evolves an existing taxonomy. Clustering-based taxonomy generation techniques (Muller *et al.*, 1999; Kashyap *et al.*, 2005; Dietz *et al.*, 2012) typically utilize hierarchical clustering and labeling techniques (Jain *et al.*, 1999) to generate taxonomy.

The TIE algorithm takes an existing taxonomy, the respective hierarchical structure (i.e., a hierarchical organization of clusters) and newly arrived documents as its input. It identifies the closest cluster for each of the newly arrived documents based on the similarity score. The range of the similarity score determines the level of impact a new document has on its closest cluster. Based on the level of impact, various reorganization operators are then applied to adjust the newly arrived document in the existing hierarchical structure. Finally, the existing taxonomy is evolved to represent the changes occurred in data. The algorithm was compared with taxonomy regeneration approach based on complexity analysis and empirical evaluation. A text data set of scholarly articles selected from computing domain was used for evaluation purpose. It was found that the incremental evolution is better than regeneration subject to time consumption. Moreover, it produces better quality taxonomy per unit time than that of regeneration of taxonomy from scratch. The main contributions of this work are summarized as follows:

- The TIE algorithm evolves an existing taxonomy whenever changes occur in data to represent updated view of the data in a shorter period of time.
- The TIE algorithm is independent of the clustering algorithm used for taxonomy generation, so existing clustering-based approaches to taxonomy generation can be effectively used with the TIE algorithm.
- A new metric named quality-time ratio is proposed to measure the effectiveness of taxonomy evolution in comparison to regeneration. It determines the ratio of taxonomy quality improvement per unit time.
- Sensitivity analysis of the TIE algorithm is done in order to determine the impact of varying different factors influencing the evolution of taxonomy.

It is declared that this paper is an extension of our earlier work (Irfan and Khan, 2016). Here the proposed algorithm is described with supplementary details. Additional evaluations in terms of the quality-time ratio and sensitivity analysis are included. The proposed algorithm is validated as a layer over both

agglomerative and divisive approaches to hierarchical clustering. The remaining paper is organized as follows: Section 2 discusses the related work. Section 3 presents the background for developing a basic understanding of the automatic taxonomy generation process. Section 4 presents the details of the proposed algorithm. Section 5 deals with the testing and evaluation of the proposed algorithm. Section 6 concludes the paper and discusses the future work.

2 Related Work

Taxonomy can be generated through various approaches, such as clustering-based (Muller *et al.*, 1999; Kashyap *et al.*, 2005; Dietz *et al.*, 2012; Neshati *et al.*, 2007; Yang *et al.*, 2015); rules and heuristics-based (Medelyan *et al.*, 2013; Lefever, 2015; Meijer *et al.*, 2014); and graph-based (Camina, 2010; Qi *et al.*, 2010; Fountain and Lapata, 2012; Velardi *et al.*, 2013) approaches. The proposed taxonomy incremental evolution (TIE) algorithm is a layer over an existing clustering-based taxonomy generation technique, so this review is limited to the clustering-based approaches. The clustering-based taxonomy generation techniques are reviewed here because of two main reasons: First of all, they contribute as a foundation in this research. Secondly, they are reviewed to identify their capabilities for handling an evolving data. In addition, other techniques that have addressed taxonomy evolution are also reviewed.

Clustering-based taxonomy generation techniques typically utilize hierarchical clustering and labeling techniques (Jain *et al.*, 1999) to generate taxonomy. The work (Muller *et al.*, 1999) is one of the pioneer attempts that utilized clustering-based approach for taxonomy generation. Their technique produced effective taxonomy with more focus on scalability so that the generated taxonomy should be less affected by the increasing size of data. Kashyap *et al.* (2005) developed an experimental framework to analyze the impact of varying different methods and parameters in the generation process of taxonomy, like the use of natural language processing versus non-natural language processing; the use of different similarity or distance measure in the clustering process; document clustering versus term clustering. Dietz *et al.* (2012) focused on the generation of domain specific taxonomy and found that general terms were not enough to construct a domain specific taxonomy. They identified domain specific concepts that were present in data, by utilizing domain specific measures of domain pertinence and domain consensus (Velardi and Sclano, 2007). Furthermore, they compared different knowledge-rich methods (i.e., based on the involvement of external knowledge sources, like Wikipedia, WordNet) and knowledge-poor methods (i.e., based on statistical and lexical properties extracted from within the data) for the extraction of relevant concepts and hierarchical relationships to see their impact on the generated taxonomy. It was observed that the clustering-based techniques involving knowledge-rich methods, to extract relevant concepts and hierarchical relationships, produced semantically better taxonomy than those techniques involving knowledge-poor methods (Neshati *et al.*, 2007). Some of the recent techniques, such as (Paukkeri *et al.*, 2012; Yang *et al.*, 2015) utilized self-organizing maps for performing clustering-based taxonomy generation. The self-organizing map is a famous artificial neural network algorithm, which is effective in mapping a high dimensional input data to a low dimension map. Each node in

the map is called neuron and it facilitates clustering by grouping together similar data objects closer in the map or under a single neuron. In short, most of the existing automatic taxonomy generation techniques are producing taxonomy effectively for small to large data sets. However, they are not focusing on handling an evolving data in taxonomy generation.

On the other hand, (Yao *et al.*, 2012) and (Marcacini and Rezende, 2010) are some of the existing works that have explored taxonomy evolution along with its generation. Yao *et al.* (2012) generated and evolved taxonomy for tag space (i.e., semi-structured data) using association rule graph (Irfan and Khan, 2016), whereas Marcacini and Rezende (2010) generated and evolved taxonomy for text data (i.e., unstructured data) using incremental hierarchical clustering-based approach (Irfan and Khan, 2016). The nature of tag data is different from unstructured text data and extracting hierarchical relationships from unstructured text data is more complex than tag data (Blumberg and Atre, 2003). This research work focuses on unstructured text data, therefore the work of Yao *et al.* (2012) was not explored further. It was observed that Marcacini and Rezende (2010) performed clustering of terms instead of document clustering. The use of term clustering for hierarchy formation is helpful because it eliminates the need to associate labels with clusters. However, document clustering is preferred because it produces more distinct clusters as compared to term clustering (Kashyap *et al.*, 2005). This work attempts to identify a solution that can incrementally evolve an existing taxonomy independent of the clustering algorithm used in the generation of taxonomy and can serve as a layer over an existing clustering-based taxonomy generation technique.

3 Background-Taxonomy Generation Process (TGP)

Since the proposed solution to taxonomy evolution can serve as a layer over an existing clustering-based taxonomy generation technique, therefore we describe the fundamentals of clustering-based taxonomy generation process (TGP) in this section. The TGP typically comprises four steps: data pre-processing, data modeling, hierarchy formation and node labeling (Irfan and Khan, 2016). We briefly describe here, the commonly used methods in these four steps for clustering-based taxonomy generation.

3.1 Data pre-processing

Data pre-processing step cleans unnecessary details from data and refines the terms that exist in the data. These refined terms reflect properties or characteristics of the data and form vocabulary of the data (Kumar and Chandrasekhar, 2012). In TGP for text data, basic natural language processing techniques, such as tokenization, stemming, part of speech tagging and parsing (Nadkarni *et al.*, 2011) have been applied to pre-process data, as in (Paukkeri *et al.*, 2012; Muller *et al.*, 1999; Spangler *et al.*, 2006; Kashyap *et al.*, 2005; Dietz *et al.*, 2012). The data, even after applying data pre-processing activities, is not in machine-readable format (i.e., a proper data model), so it is forwarded to the data modeling step for converting it into machine readable format.

3.2 Data modeling

Data modeling step finds a suitable model that expresses data in a machine-readable format for computation. Vector space model (VSM) is one of the most widely used data modeling techniques for text data (Baeza-Yates and Ribeiro-Neto, 2011) and is used in (Paukkeri *et al.*, 2012; Muller *et al.*, 1999; Spangler *et al.*, 2006; Kashyap *et al.*, 2005). For a collection of documents, VSM represents each document and its refined terms in the form of a vector that shows the occurrence of terms in the document. The occurrence of terms is represented with $tfidf$ (i.e., term frequency-inverse document frequency) score in VSM. tf, term frequency, is a number of times a particular term appears in a document; and idf, inverse document frequency, represents how often the term appears in the collection of documents. In addition, various methods are applied in data modeling step to determine the semantics of terms for their further refinement such as follows: involvement of external knowledge sources (Medelyan *et al.*, 2013), such as WordNet, Wikipedia, Freebase, DBbase; use of advanced natural language processing techniques (Dietz *et al.*, 2012), such as word sense disambiguation (WSD) (Nadkarni *et al.*, 2011); and application of dimensionality reduction techniques (Kashyap *et al.*, 2005), such as latent semantic indexing (Deerwester *et al.*, 1990). After the modeling step, data is now ready in machine-readable format for extracting hierarchical structure that exists in the data, so it is passed to the hierarchy formation step.

3.3 Hierarchy formation

Hierarchy formation step identifies and constructs a hierarchical structure inherent in data. It comprises two sub-steps: (i) relationship identification; and (ii) hierarchy generation. The relationship identification determines the relationships that exist among different documents in data. For relationship identification, similarity or distance measure, such as Cosine similarity, Euclidean distance is employed (Cha, 2007; Thada and Jaglan, 2013). The hierarchy generation arranges these relationships in the form of a hierarchical structure. Hierarchical clustering algorithms are used for hierarchy generation in clustering-based taxonomy generation techniques.

There are two types of hierarchical clustering algorithms: agglomerative (bottom up) and divisive (top down) (Jain *et al.*, 1999). The agglomerative approach starts with every data object placed in a separate cluster and merges clusters at later stages. Hierarchical agglomerative clustering (HAC) is an example of agglomerative approach and was used in (Muller *et al.*, 1999; Dietz *et al.*, 2012). Based on different merging styles, different flavors of HAC exist. The common ones are single link, complete link, average link and centroid link (Manning *et al.*, 2008). The divisive approach starts with all data objects in one cluster and divides them into more clusters at later stages. Bisect K-means clustering is an example of divisive approach and was used in (Kashyap *et al.*, 2005). Agglomerative approaches have quadratic time complexity, whereas divisive approaches have linear time complexity, nevertheless, the clustering quality is better in the case of agglomerative approaches as compared to divisive approaches (Karypis *et al.*, 2000).

Hierarchical structure, generated in this step, comprises a hierarchy of unlabeled clusters (i.e., nodes). The unlabeled hierarchical structure is forwarded to the nodes labeling step.

3.4 Nodes labeling

Nodes labeling step assigns labels to unlabeled nodes in a hierarchical structure. The assignment of meaningful and accurate labels to unlabeled nodes in a hierarchy is necessary to grasp a better understanding of generated taxonomy. In clustering-based approaches, usually centroid of a cluster (i.e., an average of all data objects in a cluster) is involved in finding labels for hierarchical nodes, as applied in (Dietz *et al.*, 2012). The labeling techniques are mostly combined with rules and heuristics in order to find appropriate labels for taxonomy (Kashyap *et al.*, 2005). External knowledge sources (Carmel *et al.*, 2009), such as WordNet; feature selection techniques (Manning *et al.*, 2008), such as mutual information, chi-square (χ^2); and frequently occurring top k terms (Glover *et al.*, 2002; Treeratpituk and Callan, 2006) have been used in literature to identify suitable labels. The output of this step is a taxonomy, where hierarchical clusters are labeled to represent a hierarchical organization of given data.

4 Proposed Method-The TIE Algorithm

In this paper, we present a detailed and an improved version of the taxonomy incremental evolution (TIE) algorithm given in (Irfan and Khan, 2016). The TIE algorithm aims at finding an incremental solution for evolving an existing taxonomy, whenever new documents are added to underlying data. The TIE algorithm is designed in a way that it serves as a layer over an existing clustering-based taxonomy generation technique. It comprises two steps:

1. Identifying the closest cluster: This step determines the closest existing cluster for each of the newly arrived documents.
2. Reorganizing the existing taxonomy: This step applies various reorganization operators to adjust a newly arrived document in the existing hierarchical structure and finally in the existing taxonomy to evolve it.

Note that the first step of the TIE algorithm, i.e., *identifying the closest cluster* is fundamentally the same as given in our earlier work (Irfan and Khan, 2016), but it now defines the actions to be taken in case a newly arrived document has more than one closest cluster; or if it is not close to any of the existing clusters. The second step, i.e., *reorganizing the existing taxonomy* now defines a new reorganization operator of $insert\_child$ for adjusting those newly arrived documents, which appear to be a part of the existing cluster but affect its quality. Next, we present the detailed explanation of these two steps of the TIE algorithm.

4.1 Identifying the closest cluster

When a new document arrives in an existing data, the TIE algorithm first identifies an appropriate cluster for adjusting the new document. The algorithm takes an existing taxonomy T_X (which is the taxonomy obtained as

a result of TGP), the respective hierarchical structure H (which is the output of the hierarchy formation step of TGP) and a set of newly arrived documents D' as its input. The hierarchy H maintains three types of information for each cluster: cluster centroid, cluster cohesion and cluster deviation.

For a cluster $c \in H$, having m document vectors: $\{\vec{d}_i\}; i = 1,2, \ldots, m$, mapped in T dimensional term space, cluster centroid $(\vec{c}_{cen})$ is the center or middle point of the cluster and is the average representation of all documents present in the cluster, given as (Irfan and Khan, 2016):

$$\vec{c}_{cen} = \frac{\sum_{i=1}^{m} \vec{d}_i}{m} \tag{1}$$

Let $sim(\vec{d}_i, \vec{c}_{cen})$ be the similarity between vectors of the i^{th} document $\vec{d}_i$ in cluster c and its centroid $\vec{c}_{cen}$, then cluster cohesion (c_μ) is the average similarity of all documents present in the cluster with the cluster centroid and is a measure of tightness of the cluster, given as (Irfan and Khan, 2016):

$$c_\mu = \frac{\sum_{i=1}^{m} sim(\vec{d}_i, \vec{c}_{cen})}{m} \tag{2}$$

Cluster deviation (c_σ) denotes the deviation of a document from a cohesive cluster. It measures the limit of closeness or farness of all documents present in the cluster from the cluster centroid, given as (Irfan and Khan, 2016):

$$c_\sigma = \sqrt{\frac{\sum_{i=1}^{m} (sim(\vec{d}_i, \vec{c}_{cen}) - c_\mu)^2}{m}} \tag{3}$$

The main idea is to identify the closest cluster for a newly arrived document. Let $sim(\vec{d}', \vec{c}_{cen})$ be the similarity between a new document $d' \in D'$ and centroid $\vec{c}_{cen}$ of the cluster c. Let us consider that c is identified as the closest cluster for d' because of maximum similarity, then the range of the similarity score determines the level of impact a new document has on its closest cluster and is defined through the following three conditions (Irfan and Khan, 2016):

1. *Far from cluster centroid:* If $sim(\vec{d}', \vec{c}_{cen}) < c_\mu - c_\sigma$, then d' is far from $\vec{c}_{cen}$. In this condition, the new document d' due to its farness from $\vec{c}_{cen}$ might affect the quality of the cluster c, thus restructuring of the cluster will be required.
2. *Close to cluster centroid:* If $sim(\vec{d}', \vec{c}_{cen}) > c_\mu + c_\sigma$, then d' is close to $\vec{c}_{cen}$. In this condition, the new document d' due to its closeness from $\vec{c}_{cen}$ might affect the labels of the cluster c (which are assumed to be dependent on the center of the cluster as in most cases for clustering-based taxonomy generation techniques), thus relabeling of the cluster will be required.
3. *Neither close to nor far from cluster centroid:* If $c_\mu - c_\sigma \leq sim(\vec{d}', \vec{c}_{cen}) \leq c_\mu + c_\sigma$, then d' is neither close to nor far from $\vec{c}_{cen}$. In this condition, the new document d' due to its presence within the range of $\vec{c}_{cen}$ might belong to the cluster c, thus it will be merged in the cluster.

This step begins by associating three types of list structures, corresponding to each of the aforementioned conditions, for each cluster in H. These list structures hold newly arrived documents based on the range of the similarity score newly arrived documents possess with their closest cluster. The list structures associated with each cluster are initialized to null. After initialization, the similarity between each of the new documents in D' and each of the existing clusters in H is calculated using the respective cluster centroid. The cluster having the maximum similarity (i.e., the closest cluster) is selected as a suitable candidate for a new document to be adjusted in, as shown in Steps 2-10 of Algorithm 1. For the cluster $c \in H$, let α be the list structure associated with the condition "far from cluster centroid"; β be the list structure associated with the condition "close to cluster centroid"; and γ be the list structure associated with the condition "neither close to nor far from cluster centroid". For a new document d' and its closest cluster c having centroid $\vec{c}_{cen}$, cohesion c_{μ} and deviation c_{σ}, the range of the similarity score of d' from $\vec{c}_{cen}$ is then checked using the aforementioned conditions and the following action is taken:

- If the condition "far from cluster centroid" is true, then add d' in the list α (see Steps 11 and 12 of Algorithm 1);
- else if the condition "close to cluster centroid" is true, then add d' in the list β (see Steps 13 and 14 of Algorithm 1);
- else if the condition "neither close to nor far from cluster centroid" is true, then add d' in the list γ (see Steps 15 and 16 of Algorithm 1).

At the end of this step, the closest cluster is identified for each of the newly arrived documents. The list structures associated with each cluster contain the newly arrived documents, based on the range of the similarity new documents possess with their closest cluster. The list structures for each cluster are the output of Algorithm 1.

In the process of determining the closest cluster for a new document, there can be two exceptions: Firstly, there is a possibility that a new document possesses equal similarity with more than one existing clusters. In that case, any of the existing clusters is randomly selected as a suitable candidate for its adjustment and appropriate actions are taken based on the range of the similarity score of that document with the centroid of the selected cluster. Another possibility is that a new document has zero (0) or no similarity with all of the existing clusters. In that case, any of the children of the root node of H is randomly selected and the new document is dealt according to the condition "far from cluster centroid".

4.2 Reorganizing the existing taxonomy

Taxonomy is a thematic representation of data. It is not necessary that the addition of a new document in the data bring drastic changes in the taxonomy. Therefore in this step, before reflecting changes in the existing taxonomy, the quality of each of the existing clusters in H is checked by recalculating the cohesion score (using existing and new documents). A higher value of cohesion represents a better cluster. Based on whether

or not a cluster quality is deteriorated and the presence of new documents in its respective list structures, reorganization operators are applied to reorganize it. Consider the cluster $c \in H$, now comprising m number of existing and m' number of new documents and let c_{sib} c_{par} and c_{chl} denote its sibling, parent and child clusters respectively, the operators applied for reorganizing the cluster are given in Table 1.

This step begins by recalculating cohesion of existing clusters in H, using existing and new documents. Thus, for the cluster $c \in H$, having m number of existing and m' number of new documents, let the new cohesion score be c'_{μ}, then there can be two possible scenarios: (i) deteriorated cluster quality; or (ii) non-deteriorated cluster quality.

Algorithm 1 Identifying the closest cluster

Input: document vectors for newly arrived documents in D', existing taxonomy T_X and existing hierarchy H with centroid, cohesion and deviation values for each cluster in H

Output: list structures for each cluster in H

initialize list structures for each cluster in H to $null$
for each $\vec{d}' \in D'$ **do**
  initialize $maximum\_similarity \leftarrow 0$; $closest\_cluster \leftarrow null$
  for each $c \in H$ **do**
    calculate $sim(\vec{d}', \vec{c}_{cen})$
    if $sim(\vec{d}', \vec{c}_{cen}) > maximum\_similarity$ **then**
      $maximum\_similarity \leftarrow sim(\vec{d}', \vec{c}_{cen})$
      $closest\_cluster \leftarrow c$
    end if
  end for
  if $maximum\_similarity < c_{\mu} - c_{\sigma}$ for the $closest\_cluster$ **then** //Steps 11 and 12 in case d' is far from $\vec{c}_{cen}$
    $\alpha \leftarrow \alpha + \vec{d}'$
  else if $maximum\_similarity > c_{\mu} + c_{\sigma}$ for the $closest\_cluster$ **then** //Steps 13 and 14 in case d' is close to $\vec{c}_{cen}$
    $\beta \leftarrow \beta + \vec{d}'$
  else if $c_{\mu} - c_{\sigma} \leq maximum\_similarity \leq c_{\mu} + c_{\sigma}$ for the $closest\_cluster$ **then** //Steps 15 and 16 in case d' is neither close to nor far from $\vec{c}_{cen}$
    $\gamma \leftarrow \gamma + \vec{d}'$
  end if
end for

Table 1 List of reorganization operators used by the TIE algorithm to reorganize the existing taxonomy

Input	Operator	Output	Function	Figure
cluster c and list α	$insert\_sibling$	sibling cluster c_{sib}	takes documents that are far from a cluster and inserts them as its sibling cluster (Irfan and Khan, 2016)	see Fig. 1a
cluster c and list γ	$insert\_child$	child cluster c_{chl}	takes documents that are neither close to nor far from a cluster and inserts them as its child cluster	see Fig. 1b
cluster c	$label\_cluster$	set of labels l_c	assigns a set of unique labels to a cluster (Irfan and Khan, 2016)	see Fig. 1c
cluster c and its set of labels l_c	$alter\_label$	set of new labels l'_c	takes a set of existing labels for a cluster and assigns a set of new and unique labels to that cluster, where the set of new and existing labels may or may not be null (Irfan and Khan, 2016)	see Fig. 1d
cluster c and list θ, which contains n number of newly arrived documents to be merged in c where, $n \leq m'$	$merge\_doc$	updated cluster c'	merges documents, contained in a given list, in a cluster (Irfan and Khan, 2016)	see Fig. 1e

updated cluster c'	$alter\_centroid$	new centroid $\vec{c}'_{cen}$	recalculates centroid for an updated cluster (Irfan and Khan, 2016)	see Fig. 1f

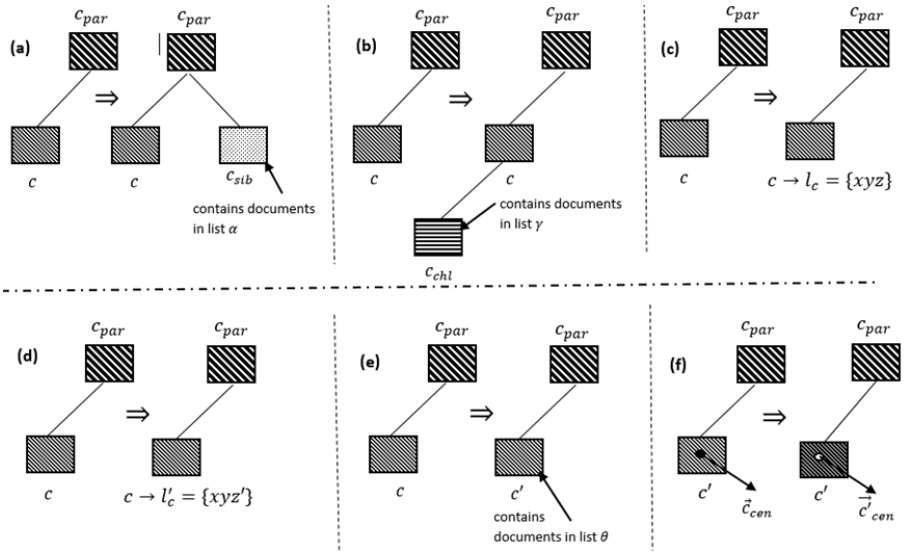


Fig. 1 Reorganization operators: (a) $c_{sib} \leftarrow insert\_sibling(c, \alpha)$ (b) $c_{chl} \leftarrow insert\_child(c, \gamma)$ (c) $l_c \leftarrow label\_cluster(c)$ (d) $l'_c \leftarrow alter\_label(c, l_c)$ (e) $c' \leftarrow merge\_doc(c, \theta)$ (f) $\vec{c}'_{cen} \leftarrow alter\_centroid(c')$

4.2.1 Scenario 1 – deteriorated cluster quality

If $c'_\mu < c_\mu$, this shows that the cluster quality is deteriorated, then based on the presence of new documents in α, β and γ lists of c possible scenarios for its reorganization are defined through the following three cases:

Super-level restructuring: Super-level restructuring occurs when a cluster quality is deteriorated and its list structure associated with "far from cluster centroid" condition is not empty. For the cluster c if $\alpha \neq null$, this indicates that documents in α which are far from the cluster centroid have resulted into its quality deterioration and should be inserted as a separate node. So $insert\_sibling$ operator is applied to create a separate node, i.e., c_{sib}, which is then assigned labels by applying $label\_cluster$ operator. This action leaves labels of parent cluster c_{par} of c as inaccurate, so it should be assigned new labels. However, before doing that the presence of documents in β and γ lists for c are also checked:

- If $\beta \neq null$ for c, then check the presence of documents in its γ list:
    - If $\gamma \neq null$, this indicates that both the lists β and γ contain some documents. So in this case, documents in β and γ first become part of c by applying $merge\_doc$ operator resulting in an updated cluster c'. The documents in γ are those that are neither close to nor far from the cluster centroid, so they can be simply merged in the cluster (assuming to have minimal effect on its quality). However, the documents in β are those that are close to the cluster centroid. The centroid is no more an accurate representation of that cluster and this requires recalculation of the cluster centroid and assignment of new

labels. So $alter\_centroid$ operator is applied to recalculate c' centroid using existing and new documents. Finally, new labels are assigned first to c' and then to c_{par} by applying $alter\_label$ operator.

- If $\gamma = null$, this indicates that the list β contains some documents, but the list γ is empty. So in this case, only documents in β first become part of c by applying $merge\_doc$ operator resulting in an updated cluster c'. The documents in β are those that are close to the cluster centroid. The centroid is no more an accurate representation of that cluster and this requires recalculation of the cluster centroid and assignment of new labels. So $alter\_centroid$ operator is applied to recalculate c' centroid using existing and new documents. Finally, new labels are assigned first to c' and then to c_{par} by applying $alter\_label$ operator.

- If $\beta = null$ for c, then check the presence of documents in its γ list:
  - If $\gamma \neq null$, this indicates that though the list β is empty, but the list γ contains some documents. So in this case, only documents in γ first become part of c by applying $merge\_doc$ operator resulting in an updated cluster c'. The documents in γ are those that are neither close to nor far from the cluster centroid, so they can be simply merged in the cluster (assuming to have minimal effect on its quality) and therefore no need to recalculate centroid and assign new labels to the updated cluster. New labels are assigned only to c_{par} by applying $alter\_label$ operator.
  - If $\gamma = null$, this indicates that both the lists β and γ are empty and no new document is there in these lists to add in cluster c. So in this case, without taking any further actions new labels are assigned to c_{par} by applying $alter\_label$ operator.

The actions taken for the super-level restructuring are shown in Steps 4-26 of Algorithm 2.

Relabeling: Relabeling occurs when a cluster quality is deteriorated and its list structure associated with "far from cluster centroid" condition is empty, but the one associated with "close to cluster centroid" condition contains some documents. For the cluster c if $\alpha = null$ and $\beta \neq null$, this indicate that documents in β that are close to the cluster centroid have affected its centroid representation. So it should be recalculated, followed by the assignment of new labels to the cluster. However, before doing that the presence of documents in γ list for c is also checked:

- If $\gamma \neq null$, this indicates that the list γ also contains some documents along with the list β. This case is similar to the super-level restructuring case when the β and γ lists are not empty along with the α list, so the same set of reorganization operators are applied to perform the relabeling. First, the documents in both the lists β and γ become part of c by applying $merge\_doc$ operator resulting in an updated cluster c'. After that, $alter\_centroid$ operator is applied to recalculate c' centroid using existing and new documents. New labels are then assigned to c' by applying

$alter\_label$ operator. This action leaves labels of parent cluster c_{par} of c' as inaccurate, so new labels are assigned to c_{par} by applying $alter\_label$ operator.

- If $\gamma = null$, this indicates that the list γ is empty. This case is similar to the super-level restructuring case when the β list is not empty along with the α list but the γ list is empty, so the same set of reorganization operators are applied to perform the relabeling. First, the documents in β list become part of c by applying $merge\_doc$ operator resulting in an updated cluster c'. After that $alter\_centroid$ operator is applied to recalculate c' centroid using existing and new documents. New labels are then assigned to c′ by applying $alter\_label$ operator. This action leaves labels of parent cluster c_{par} of c' as inaccurate, so new labels are assigned to c_{par} by applying $alter\_label$ operator.

The actions taken for the relabeling are shown in Steps 27-38 of Algorithm 2.

Sub-level restructuring: Sub-level restructuring occurs when a cluster quality is deteriorated and its list structures associated with "far from cluster centroid" and "close to cluster centroid" conditions are empty, but the one associated with "neither close to nor far from cluster centroid" condition contains some documents. For the cluster c if $\alpha = null$, $\beta = null$ and $\gamma \neq null$, this indicate that documents in γ that are neither close to nor far from the cluster centroid can be simply merged in the cluster (assuming to have minimal effect on its quality). But in this case, both α and β lists are empty and only the documents in γ are affecting cluster quality, so instead of simply merging them in the cluster, we create a child node c_{chl} by applying $insert\_child$ operator and insert the documents in the list γ in it. The operator $label\_cluster$ is then applied to assign labels to c_{chl}. This action leaves labels of c as inaccurate, so new labels are assigned to c by applying $alter\_label$ operator.

The actions taken for the sub-level restructuring are shown in Steps 39-43 of Algorithm 2.

Example scenario 1 – deteriorated cluster quality: For a cluster $cx \in H$, let's assume that α_{cx}, β_{cx} and γ_{cx} be associated with the conditions "far from cluster centroid", close to cluster centroid" and "neither close to nor far from cluster centroid" respectively and contain document vectors for newly arrived documents $(d'_1, d'_2), (d'_3)$ and (d'_4) respectively. For the cluster cx, let's further assume that its quality when calculated with new and existing documents was found to be deteriorated, i.e., $cx'_\mu < cx_\mu$. As per the steps shown in Algorithm 2, first α_{cx} list is checked for the presence of documents. As $\alpha_{cx} \neq null$, so super-level restructuring (i.e., Steps 4-26) is applied here. The documents d'_1 and d'_2 are inserted as a sibling cluster of cx by applying $insert\_sibling$ operator and labels are assigned to the sibling cluster by applying $label\_cluster$ operator. After that β_{cx} list is checked for the presence of documents. Here $\beta_{cx} \neq null$, but before applying reorganization operators for $\beta_{cx} \neq null$, γ_{cx} list is also checked. As in this case $\gamma_{cx} \neq null$, so $merge\_doc$ operator is applied to merge documents present in both β_{cx} and γ_{cx} lists. The documents d'_3 and d'_4 present in β_{cx} and γ_{cx} respectively become part of cx. Since the cluster cx is updated, let's denote the updated cluster as cx'. Note that we check γ_{cx} list before applying operators associated with β_{cx} list due to the fact that if centroid

recalculation and relabeling would be done before merging all the possible documents, then they would not accurately represent the cluster. As γ_{cx} list is checked and appropriate actions are already taken, so for cx' centroid is recalculated using $alter\_centroid$ operator. Soon after altering the centroid, the updated cluster is assigned new labels by applying $alter\_label$ operator. The addition of the sibling cluster and the updation of cx affect the label of their parent cluster, so $alter\_centroid$ operator is applied to assign new labels to the parent cluster.

4.2.2 Scenario 2 – non-deteriorated cluster quality

If $c'_\mu > c_\mu$, then updates are applied to the cluster through the following simple case:

Simple merging: Simple merging occurs when a cluster quality is not deteriorated due to the addition of new documents. For the cluster c, this case simply merges documents in its α, β and γ lists in the cluster by applying $merge\_doc$ operator, resulting in an updated cluster c', as shown in Steps 44-46 of Algorithm 2.

Example scenario 2 – non-deteriorated cluster quality: For a cluster $cy \in H$, let's assume that α_{cy}, β_{cy} and γ_{cy} be associated with the conditions "far from cluster centroid", close to cluster centroid" and "neither close to nor far from cluster centroid" respectively and contain document vectors for newly arrived documents $(d'_5), (\emptyset)$ and (d'_6) respectively. For the cluster cy, let's further assume that its quality when calculated with new and existing documents was found not to be deteriorated, i.e., $cy'_\mu > cy_\mu$. This is a simple merging (i.e., Steps 44-46) case and all the newly arrived documents are simply merged in the cluster. This is because the possible existence of newly arrived documents is not causing deterioration in the cluster quality. It would be time and cost efficient to simply merge the new documents in the cluster without making any adjustments, like centroid recalculation or relabeling the cluster. So as per the steps shown in Algorithm 2, the newly arrived documents, i.e., d'_5 and d'_6 are simply merged in the cluster cy by applying $merge\_doc$ operator.

The output of this step of the TIE algorithm is an evolved version of the existing taxonomy.

Algorithm 2 Reorganizing the existing taxonomy

Input: list structures and cohesion for each cluster in H
Output: evolved taxonomy T'

1. **for each** $c \in H$ **do**
2. calculate c'_μ for c using m existing and m' new documents
3. **if** $c'_\mu < c_\mu$ **then**
4. **if** $\alpha \neq null$ **then** //Steps 4-26: *Super-level restructuring*
5. apply $c_{sib} \leftarrow insert\_sibling(c, \alpha)$
6. apply $l_{c_{sib}} \leftarrow label\_cluster(c_{sib})$
7. **if** $\beta \neq null$ then
8. **if** $\gamma \neq null$ then
9. apply $c' \leftarrow merge\_doc(c, \beta + \gamma)$
10. apply $\vec{c}'_{cen} \leftarrow alter\_centroid(c')$
11. apply $l'_{c'} \leftarrow alter\_label(c', l_{c'})$
12. apply $l'_{c_{par}} \leftarrow alter\_label(c_{par}, l_{c_{par}})$ //c_{par}: parent node
13. **else**
14. apply $c' \leftarrow merge\_doc(c, \beta)$
15. apply $\vec{c}'_{cen} \leftarrow alter\_centroid(c')$

16. apply $l'_{c'} \leftarrow alter\_label(c', l_{c'})$
17. apply $l'_{c_{par}} \leftarrow alter\_label(c_{par}, l_{c_{par}})$
18. **end if**
19. **else**
20. **if** $\gamma \neq null$ **then**
21. apply $c' \leftarrow merge\_doc(c, \gamma)$
22. apply $l'_{c_{par}} \leftarrow alter\_label(c_{par}, l_{c_{par}})$
23. **else**
24. apply $l'_{c_{par}} \leftarrow alter\_label(c_{par}, l_{c_{par}})$
25. **end if**
26. **end if**
27. **else if** $\beta \neq null$ **then** //Steps:27-38: *Relabeling*
28. **if** $\gamma \neq null$ **then**
29. apply $c' \leftarrow merge\_doc(c, \beta + \gamma)$
30. apply $\vec{c}'_{cen} \leftarrow alter\_centroid(c')$
31. apply $l'_{c'} \leftarrow alter\_label(c', l_{c'})$
32. apply $l'_{c_{par}} \leftarrow alter\_label(c_{par}, l_{c_{par}})$
33. **else**
34. apply $c' \leftarrow merge\_doc(c, \beta)$
35. apply $\vec{c}'_{cen} \leftarrow alter\_centroid(c')$
36. apply $l'_{c'} \leftarrow alter\_label(c', l_{c'})$
37. apply $l'_{c_{par}} \leftarrow alter\_label(c_{par}, l_{c_{par}})$
38. **end if**
39. **else if** $\gamma \neq null$ **then** //Steps 39-43: *Sub-level restructuring*
40. apply $c_{chl} \leftarrow insert\_child(c, \gamma)$ //c_{chl}: child node
41. apply $l_{c_{chl}} \leftarrow label\_cluster(c_{chl})$
42. apply $l'_c \leftarrow alter\_label(c, l_c)$
43. **end if**
44. **else** //Steps 44-46: *Simple merging*
45. apply $c' \leftarrow merge\_doc(c, \alpha + \beta + \gamma)$
46. **end if**
47. **end for**

4.3 Complexity analysis

This subsection presents complexity analysis to check the independency of the TIE algorithm on the clustering algorithm used for the generation of taxonomy. Moreover, the time efficiency of taxonomy evolution in comparison to taxonomy regeneration is also checked through the complexity analysis. Let n denote a number of documents in a data set. The time complexity of HAC algorithm (agglomerative approach) is given as a function of $n \times n$ distance/similarity matrix, i.e., n^2 (Karypis *et al.*, 2000). The time complexity of bisect K-means algorithm (divisive approach) is given as a function of partition parameter and a number of documents to partition. Let the partition parameter be denoted as K and a number of documents to partition be denoted as n, so we can say that the time complexity of bisect K-means is Kn (Karypis *et al.*, 2000). Let x denote a number of hierarchical clusters formed as a result of applying hierarchical clustering (HAC or bisect K-means), where logically we assume that $x \ll n$. Let n' denote a number of newly arrived documents, where for shorter time intervals we assume that $n' \ll n$.

Time complexity of TGP: As hierarchy formation is a major step involved in the generation of taxonomy, so we ignore the cost associated with other steps of taxonomy generation. Let the time complexity

of TGP in the case of HAC and bisect K-means be denoted as $\tau_{TGP}(h)$ and $\tau_{TGP}(b)$ respectively and they are approximated as:

$$\tau_{TGP}(h) \approx n^2 \quad (4)$$

$$\tau_{TGP}(b) \approx Kn \quad (5)$$

Time complexity of TIE: For the TIE algorithm, we can say that for each newly arrived document, x comparisons are made to identify the closest cluster. If we consider ignoring the time taken in applying the reorganization operators because of the linear trend, then let the time complexity of TIE in the case of HAC and bisect K-means be denoted as $\tau_{TIE}(h)$ and $\tau_{TIE}(b)$ respectively and they are approximated as:

$$\tau_{TIE}(h) \approx xn' \quad (6)$$

$$\tau_{TIE}(b) \approx xn' \quad (7)$$

TIE is independent of the clustering algorithm used in TGP: From Equation (6) and Equation (7), we can see that the time complexity of TIE algorithm in the case of both the clustering algorithms (i.e., HAC and bisect K-means) is same. This shows that the TIE algorithm is independent of the clustering algorithm used for the generation of taxonomy and serves as a layer over any of the existing clustering-based taxonomy generation techniques.

Evolution versus Regeneration: We can approximate the time of regeneration by using Equation (4) and Equation (5) in the case of HAC and bisect K-means respectively. Let the time complexity of regeneration in the case of HAC and bisect K-means be denoted as $\tau_r(h)$ and $\tau_r(b)$ and they are given as:

$$\tau_r(h) \approx n^2 + n^2 \quad (8)$$

$$\tau_r(b) \approx Kn + Kn \quad (9)$$

On the other hand, we can approximate the time of evolution by combining Equation (4) and Equation (6) in the case of HAC, and by combining Equation (5) and (7) in the case of bisect K-means. Let the time complexity of evolution in the case of HAC and bisect K-means be denoted as $\tau_e(h)$ and $\tau_e(b)$ and they are given as:

$$\tau_r(h) \approx n^2 + xn' \quad (10)$$

$$\tau_r(b) \approx Kn + xn' \quad (11)$$

In the case of HAC using Equation (8) and Equation (10), we can observe that $n^2 + xn' \ll n^2 + n^2$ (as $x, n' \ll n$), so taxonomy evolution time is much less than that of taxonomy regeneration. In the case of bisect K-means using Equations (9) and Equation (11), we can observe that $Kn + xn' \ll Kn + Kn$ (as $x, n' \ll n$), so taxonomy evolution time is much less than that of taxonomy regeneration.

5 Evaluation

The taxonomy incremental evolution was evaluated in comparison to taxonomy regeneration from scratch on a text data set of scholarly articles selected from computing domain. The details of the data set, evaluation metrics, experiments and their results are given next.

5.1 Data set specification

For computing domain, four hundred (400) scholarly articles in pdf format were randomly downloaded from ACM Digital Library (http://dl.acm.org/). These documents were then converted into text format using Apache Tika (https://tika.apache.org/). ACM Computing Classification System (CCS) (http://www.acm.org/about/class/2012), which is a standard topic hierarchy for computing domain, was adopted as a reference taxonomy.

5.2 Metrics

The taxonomies obtained through evolution and regeneration approaches underwent three types of evaluation: (i) time-based, (ii) quality-based, and (iii) quality and time-based. The first two of these evaluation methods are the same as adopted in our earlier work (Irfan and Khan, 2016), but the last one is the newly proposed method.

5.2.1 Time-based evaluation

The time-based evaluation was done by observing the runtime of applications associated with both the approaches (Irfan and Khan, 2016). Let it denote as t_{run}.

5.2.2 Quality-based evaluation

The quality-based evaluation was based on taxonomy's lexical and hierarchical quality. The lexical quality assesses the quality of taxonomy labels, whereas the hierarchical quality assesses the quality of hierarchical associations among the taxonomic nodes. For the lexical quality assessment, we adopted lexical precision (LP), lexical recall (LR) and lexical F-measure (LF), which have been used in many existing taxonomy generation techniques (Dietz *et al.*, 2012; Cimiano *et al.*, 2005). The lexical quality matches a label in both computed and reference taxonomies. No matter what kind of relationship a label possesses with other labels in both the taxonomies, it is considered as a match. We assessed the lexical quality of both the evolved and regenerated taxonomies, in comparison to reference taxonomy, using these metrics. For the hierarchical quality assessment, we adopted hierarchical precision (HP), hierarchical (HR) and hierarchical F-measure (HF) proposed in (Irfan and Khan, 2016). For a label that appears in both computed and reference taxonomies, the hierarchical quality compares its parent's and ancestors' labels in both the taxonomies to identify a match. The formulas and the definitions for the lexical and the hierarchical quality metrics are given in Tables 2 and 3 respectively.

5.2.3 Quality and time-based evaluation

In this work, we propose a new metric for assessing the efficiency of taxonomy evolution in comparison to taxonomy regeneration. The quality-time ratio combines the quality-based and time-based metrics and is the rate of improvement in taxonomy quality per unit time. Let the quality-time ratio be denoted as r_{QT} and is defined as:

$$r_{QT} = \frac{\sum Quality\ based\ metrics}{t_{run}} \tag{12}$$

The r_{QT} value is calculated by summing up the values of both the lexical and the hierarchical quality metrics to determine the overall improvement in taxonomy quality per unit time (i.e., the runtime), in the case of evolution and regeneration. Although there is no limit on the range of values for this measure, however in our case as the quality-based metrics values range between 0 and 1 and time (measured in minutes (denoted as $mins$)) is greater than 1 in all the cases, so we are getting the values in the range of 0-1.

Table 2 Lexical quality metrics

Lexical quality metrics		
Given that L_C= a set of all labels in computed (regenerated/evolved) taxonomy; L_R= a set of all labels in reference taxonomy		
Measure	Formula	Description
Lexical precision (LP)	$LP = \frac{\lvert L_C \cap L_R \rvert}{\lvert L_C \rvert}$	calculates the percentage of labels in computed (regenerated/evolved) taxonomy that are also there in reference taxonomy (Irfan and Khan, 2016)
Lexical recall (LR)	$LR = \frac{\lvert L_C \cap L_R \rvert}{\lvert L_R \rvert}$	calculates the percentage of labels in reference taxonomy that are also there in computed (regenerated/evolved) taxonomy (Irfan and Khan, 2016)
Lexical F-measure (LF)	$HF = \frac{2(LP \times LR)}{(LP + LR)}$	the harmonic mean of lexical precision and recall (Irfan and Khan, 2016)

Table 3 Hierarchical quality metrics

Hierarchical quality metrics		
Given that $L_C \cap L_R$ = a set of common labels in computed (regenerated/evolved) taxonomy and reference taxonomy; $par_{a_i}[T']$ = a set of all parent associations and $anc_{a_i}[T']$= a set of all ancestor associations for an i^{th} term a_i in T' (where, T' can be computed (regenerated/evolved) taxonomy, i.e., T_c or reference taxonomy, i..e, T_R) and note that $par_{a_i}[T'] \subseteq anc_{a_i}[T']$		
Measure	Formula	Description
Hierarchical precision (HP)	$HP = \frac{\sum_{i=1}^{\lvert L_C \cap L_R \rvert} \lvert par_{a_i}[T_C] \cap anc_{a_i}[T_R] \rvert}{\sum_{i=1}^{\lvert L_C \cap L_R \rvert} \lvert par_{a_i}[T_C] \rvert}$	calculates the percentage of all parent associations in computed (regenerated/evolved) taxonomy that also appear in reference taxonomy as parent or ancestor associations (Irfan and Khan, 2016)
Hierarchical recall (HR)	$HR = \frac{\sum_{i=1}^{\lvert L_C \cap L_R \rvert} \lvert par_{a_i}[T_R] \cap anc_{a_i}[T_C] \rvert}{\sum_{i=1}^{\lvert L_C \cap L_R \rvert} \lvert par_{a_i}[T_R] \rvert}$	calculates the percentage of all parent associations in reference taxonomy that also appear in computed (regenerated/evolved) taxonomy as parent or ancestor associations (Irfan and Khan, 2016)
Hierarchical F-measure (HF)	$HF = \frac{2(HP \times HR)}{(HP + HR)}$	the harmonic mean of hierarchical precision and recall (Irfan and Khan, 2016)

5.3 Experiments

The experiments performed were divided into three parts: Firstly, an initial taxonomy was generated using a base data set to be adopted as a foundation for performing regeneration and evolution. Secondly, with the addition of new documents taxonomy regeneration was performed for different size data sets to measure the

quality and the time metrics values. Finally, with the addition of new documents taxonomy evolution was performed for different size data sets to measure the quality and the time metrics values. The details of the experiments performed are given next.

5.3.1 Initial taxonomy generation

Since the main purpose was to check the improvement of evolution over regeneration rather than the improvement in taxonomy generation process, therefore we adopted a simple approach towards the generation of taxonomy. The type of the data set is textual, therefore we adopted methods suitable for taxonomy generation process (TGP) of text data. Note that the TGP adopted is very much similar to the one adopted in our earlier work (Irfan and Khan, 2016), with the exception of adopting two different types of hierarchical clustering approaches, i.e., agglomerative and divisive, so that the independency of the TIE algorithm on the clustering algorithm used in the taxonomy generation process can be checked empirically. TGP was applied on a set of 200 randomly selected documents to generate an initial taxonomy. Following are the steps involved in generating an initial taxonomy:

1. TGP performed natural language processing of tokenization, stemming, stop word removal and parsing to extract nounphrases. These nounphrases formed the vocabulary of distinct terms for the given data set.
2. Afterward, vector space modeling was applied to express the vocabulary and the data set in the form of a vector model.
3. In order to evaluate independency of the TIE algorithm on the clustering algorithm used in TGP, we performed experiments with two types of hierarchical clustering algorithms: HAC (agglomerative approach) and bisect K-means (divisive approach). For HAC, we implemented three different variants, i.e., single link, complete link and average link to select the best one. It was found through various test runs that the average link produced the best results for generating taxonomy as compared to other variants of HAC. The experimental results for different variants of HAC presented in Table 4 show the average precision, recall and F-measure (lexical) values along with the standard deviations (σ) for the generated taxonomy on the data set of 200 documents. After this step, we had two hierarchical structures (i.e., hierarchical organization of unlabeled clusters) each obtained from HAC (average link) and bisect K-means and the evaluation was performed for the taxonomies obtained from each of these hierarchical structures. Moreover, for bisect K-means, we approximated the value of $K \approx \sqrt{\frac{n}{2}}$ (as K should be $\ll n$ (Karypis *et al.*, 2000)) where n is a number of documents to partition.
4. For labeling the hierarchical structures produced through both of the hierarchical clustering algorithms, top 25 terms (based on $tfidf$ score) of the cluster centroid were adopted as labels and they were further pruned to keep general terms as labels for parent clusters and specific terms as

labels for child clusters. The reason for choosing 25 terms to label a cluster is that too few or too many terms make the cluster (i.e., taxonomic node) hard to interpret. This step produced two taxonomies, each corresponding to one of the two hierarchical structures.

Table 4 Comparison of different variants of HAC based on lexical quality metrics

For a data set of 200 documents			
HAC variants	LP (average $\pm \sigma$)	LR (average $\pm \sigma$)	LF
Average link HAC	0.580 ($\pm$0.010)	0.457 ($\pm$0.044)	0.511
Complete link HAC	0.437($\pm$0.019)	0.347($\pm$0.072)	0.387
Single link HAC	0.319($\pm$0.006)	0.234 ($\pm$0.010)	0.270

5.3.2 Taxonomy regeneration

In order to evaluate the regeneration of taxonomy from scratch when new documents are added to data set, the set of 200 randomly selected documents used to generate the initial taxonomy was adopted as a base data set. Two sets of experiments were performed: one for evaluating the regeneration of taxonomy in the case of HAC, and the other for evaluating the regeneration of taxonomy in the case of bisect K-means. For the case of HAC, first 20 documents were added to the data set of 200 documents and the TGP using HAC was performed from scratch for 220 documents. As a result of which the initial taxonomy of 200 documents was replaced with the new taxonomy for 220 documents. After that, 30 more documents were added to the data set of 220 documents and the TGP using HAC was performed from scratch for 250 documents. As a result of which the existing taxonomy of 220 documents was replaced with the new taxonomy for 250 documents. Experiments like this were repeated by adding 40, 50 and 60 more documents in the data set of 250, 290 and 340 documents respectively. Each of the regenerated taxonomies was evaluated in comparison to ACM CSS to get the results for the lexical and the hierarchical quality metrics, as well as the runtime. The same set of experiments were repeated for evaluating the regeneration of taxonomy in the case of bisect K-means.

5.3.3 Taxonomy evolution

In order to evaluate the evolution of taxonomy, similar set of experiments were performed as in the case of taxonomy regeneration (see Section 5.3.2). Starting off with the initial taxonomy of 200 documents obtained using HAC, new documents (in the set of 20, 30, 40, 50 and 60) were added in the system and the TIE algorithm was run for different size data sets. The lexical and the hierarchical quality metrics, as well as the runtime of the taxonomy evolved as a result of each run of the TIE algorithm were noted. The same set of experiments were repeated for evaluating the evolution of taxonomy in the case of bisect K-means.

5.4 Experimental results

The runtime (i.e., t_{run}), the lexical quality metrics (i.e., LP, LR and LF) and the hierarchical quality metrics (i.e., HP, HR and HF) obtained as a result of taxonomy regeneration (see Section 5.3.2) and taxonomy evolution (see Section 5.3.3) in the case of HAC are shown in Tables 5–7 respectively, whereas the runtime

(i.e., t_{run}), the lexical quality metrics (i.e., LP, LR and LF) and the hierarchical quality metrics (i.e., HP, HR and HF) obtained as a result of taxonomy regeneration (see Section 5.3.2) and taxonomy evolution (see Section 5.3.3) in the case of bisect K-means are shown in Tables 8–10 respectively. We calculated the average values for each measure by performing several runs of the experiments and randomly selecting the document sets for each run. All results mentioned in Tables 5–10 are stating the average values along with the standard deviations (σ).

Table 5 Results of time-based evaluation (in the case of HAC)

With HAC	t_{run} $(\pm\sigma)$ $(mins)$	
Data set	TGP	TIE
200	22.842 (±0.308)	–
200 + 20 = 220	24.725 (±0.080)	4.905 (±0.232)
220 + 30 = 250	28.030(±0.170)	7.182 (±0.245)
250 + 40 = 290	46.760 (±0.009)	9.810 (±0.180)
290 + 50 = 340	59.390 (±0.140)	15.601 (±0.238)
340 + 60 = 400	70.088 (±0.437)	20.675 (±0.337)

Table 6 Results of lexical quality-based evaluation (in the case of HAC)

With HAC	LP (average $\pm\sigma$)		LR (average $\pm\sigma$)		LF	
Data set	TGP	TIE	TGP	TIE	TGP	TIE
200	0.580 (±0.010)	–	0.457 (±0.004)	–	0.511	–
200 + 20 = 220	0.581 (±0.020)	0.522 (±0.008)	0.440 (±0.001)	0.425 (±0.013)	0.501	0.469
220 + 30 = 250	0.539 (±0.017)	0.516 (±0.010)	0.434 (±0.004)	0.415 (±0.035)	0.481	0.460
250 + 40 = 290	0.532 (±0.009)	0.501 (±0.065)	0.424 (±0.002)	0.402 (±0.056)	0.472	0.446
290 + 50 = 340	0.512 (±0.042)	0.499 (±0.061)	0.409 (±0.002)	0.392 (±0.014)	0.455	0.439
340 + 60 = 400	0.507 (±0.004)	0.490 (±0.061)	0.411 (±0.003)	0.401 (±0.045)	0.454	0.441

Table 7 Results of hierarchical quality-based evaluation (in the case of HAC)

With HAC	HP (average $\pm\sigma$)		HR (average $\pm\sigma$)		HF	
Data set	TGP	TIE	TGP	TIE	TGP	TIE
200	0.351 (±0.099)	–	0.330 (±0.010)	–	0.340	–
200 + 20 = 220	0.341 (±0.007)	0.338 (±0.006)	0.323 (±0.010)	0.283 (±0.001)	0.331	0.308
220 + 30 = 250	0.346 (±0.068)	0.307 (±0.004)	0.319 (±0.002)	0.275 (±0.013)	0.332	0.290
250 + 40 = 290	0.326 (±0.004)	0.304 (±0.020)	0.298 (±0.033)	0.277 (±0.003)	0.311	0.292
290 + 50 = 340	0.312 (±0.003)	0.295 (±0.003)	0.295 (±0.003)	0.276 (±0.004)	0.304	0.285
340 + 60 = 400	0.305 (±0.009)	0.291 (±0.086)	0.286 (±0.006)	0.264 (±0.003)	0.295	0.277

Table 8 Results of time-based evaluation (in the case of bisect K-means)

With bisect K-means	t_{run} $(\pm\sigma)$ $(mins)$	
Data set	TGP	TIE
200	21.157 (±0.012)	–
200 + 20 = 220	22.569 (±0.310)	4.438 (±0.010)
220 + 30 = 250	23.981 (±0.048)	6.479 (±0.011)
250 + 40 = 290	30.642 (±0.211)	10.911 (±0.020)

$290 + 50 = 340$	$45.093\ (\pm 0.224)$	$13.714\ (\pm 0.014)$
$340 + 60 = 400$	$57.503\ (\pm 0.126)$	$19.704\ (\pm 0.050)$

Table 9 Results of lexical quality-based evaluation (in the case of bisect K-means)

With bisect K-means	LP (average $\pm\ \sigma$)		LR (average $\pm\ \sigma$)		LF	
Data set	TGP	TIE	TGP	TIE	TGP	TIE
200	$0.445\ (\pm 0.007)$	–	$0.323\ (\pm 0.002)$	–	0.374	–
$200 + 20 = 220$	$0.405\ (\pm 0.009)$	$0.380\ (\pm 0.015)$	$0.338\ (\pm 0.086)$	$0.327\ (\pm 0.020)$	0.369	0.352
$220 + 30 = 250$	$0.402\ (\pm 0.010)$	$0.354\ (\pm 0.012)$	$0.351\ (\pm 0.005)$	$0.323\ (\pm 0.002)$	0.375	0.338
$250 + 40 = 290$	$0.408\ (\pm 0.011)$	$0.358\ (\pm 0.001)$	$0.317(\pm 0.003)$	$0.292\ (\pm 0.016)$	0.356	0.322
$290 + 50 = 340$	$0.395\ (\pm 0.012)$	$0.354\ (\pm 0.012)$	$0.311\ (\pm 0.094)$	$0.310\ (\pm 0.002)$	0.347	0.331
$340 + 60 = 400$	$0.390\ (\pm 0.037)$	$0.338\ (\pm 0.004)$	$0.304\ (\pm 0.083)$	$0.296(\pm 0.111)$	0.342	0.316

Table 10 Results of hierarchical quality-based evaluation (in the case of bisect K-means)

With bisect K-means	HP (average $\pm\ \sigma$)		HR (average $\pm\ \sigma$)		HF	
Data set	TGP	TIE	TGP	TIE	TGP	TIE
200	$0.244\ (\pm 0.089)$	–	$0.217\ (\pm 0.054)$	–	0.229	–
$200 + 20 = 220$	$0.241\ (\pm 0.011)$	$0.217\ (\pm 0.013)$	$0.228\ (\pm 0.032)$	$0.198\ (\pm 0.011)$	0.234	0.207
$220 + 30 = 250$	$0.242\ (\pm 0.008)$	$0.202\ (\pm 0.008)$	$0.220\ (\pm 0.051)$	$0.177\ (\pm 0.052)$	0.231	0.189
$250 + 40 = 290$	$0.233\ (\pm 0.002)$	$0.197\ (\pm 0.004)$	$0.204\ (\pm 0.076)$	$0.203\ (\pm 0.095)$	0.218	0.199
$290 + 50 = 340$	$0.218\ (\pm 0.031)$	$0.196\ (\pm 0.002)$	$0.193\ (\pm 0.007)$	$0.185\ (\pm 0.065)$	0.205	0.190
$340 + 60 = 400$	$0.208\ (\pm 0.053)$	$0.179\ (\pm 0.014)$	$0.188\ (\pm 0.006)$	$0.162\ (\pm 0.034)$	0.197	0.170

The time and the quality metrics values listed in Tables 5–10 were used to calculate the quality-time ratio (i.e., r_{QT}), whose graphs are shown in Figs. 2–4 in the case of HAC and in Figs. 5–7 in the case of bisect K-means.

Moreover, it was observed that a number of documents used to form the existing taxonomy and a number of newly added documents in the data set are the two factors that can mainly influence the evolution of taxonomy, let them denote as $f1$ and $f2$ respectively. So we performed sensitivity analysis to check the impact of varying these factors on the evolution of taxonomy. Two sets of additional experiments were performed for this purpose: firstly, taxonomy evolution was performed for different size data sets by varying $f1$ and keeping $f2$ fixed; secondly, taxonomy evolution was performed for different size data sets by keeping $f1$ fixed and varying $f2$. The runtime (i.e., t_{run}), the lexical quality metrics (i.e., LP, LR and LF) and the hierarchical quality metrics (i.e., HP, HR and HF) values obtained as a result of these experiments are shown in Tables 11–14. Like Tables 5–10, the values in Tables 11–14 are also based on several runs of the experiments and are stating the average values along with the standard deviations (σ).

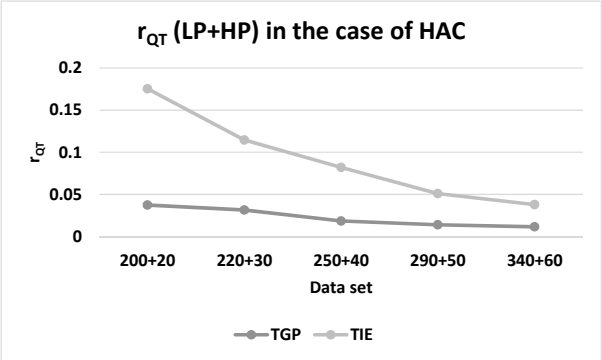


Fig. 2 Graph for $r_{QT}(LP + HP)$ in the case of HAC

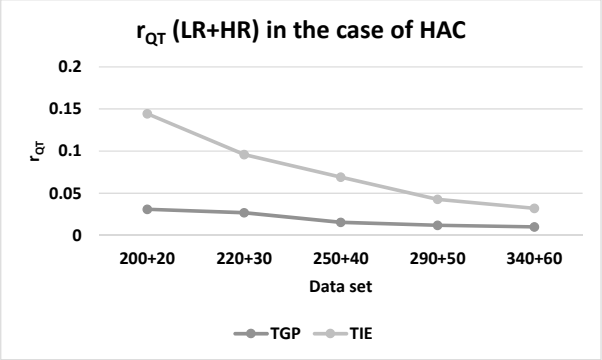


Fig. 3 Graph for $r_{QT}(LR + HR)$ in the case of HAC

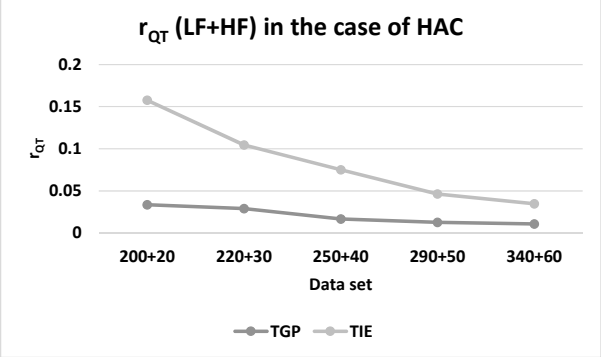


Fig. 4 Graph for $r_{QT}(LF + HF)$ in the case of HAC

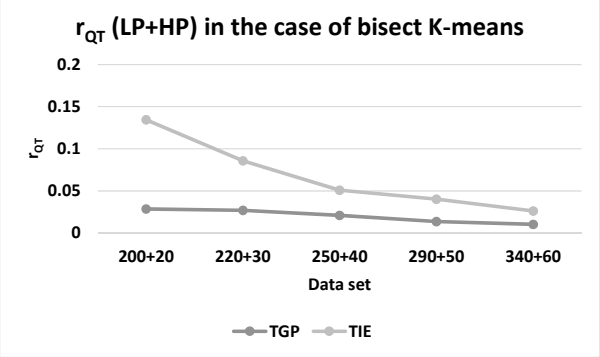
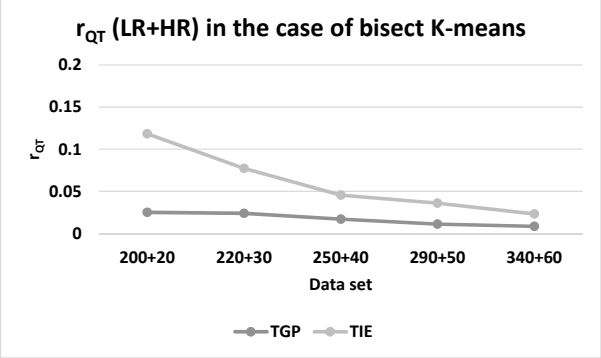


Fig. 5 Graph for $r_{QT}(LP + HP)$ in the case of bisect K-means



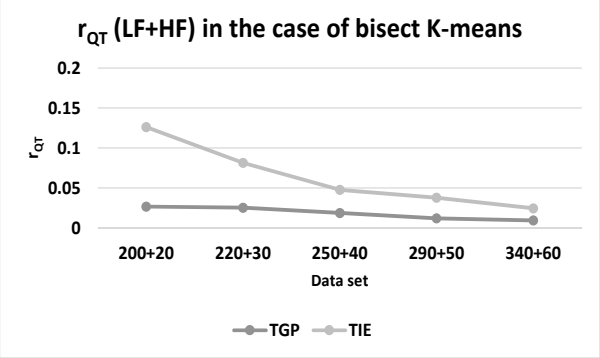


Fig. 6 Graph for $r_{QT}(LR + HR)$ in the case of bisect K-means

Fig. 7 Graph for $r_{QT}(LF + HF)$ in the case of bisect K-means

Table 11 Sensitivity analysis of the TIE algorithm by varying $f1$ and keeping $f2$ fixed (in the case of HAC)

$f1$	$f2$	Data set	t_{run} $(\pm\sigma)$ $(mins)$	LP (average $\pm\sigma$)	LR (average $\pm\sigma$)	LF	HP (average $\pm\sigma$)	HR (average $\pm\sigma$)	HF
200	30	230	5.024 (±0.093)	0.528 (±0.017)	0.438 (±0.312)	0.479	0.355 (±0.101)	0.326 (±0.088)	0.340
230	30	260	7.298 (±0.028)	0.526 (±0.003)	0.430 (±0.077)	0.473	0.348 (±0.002)	0.322 (±0.011)	0.334
260	30	290	10.016 (±0.005)	0.519 (±0.015)	0.424 (±0.008)	0.467	0.346 (±0.014)	0.319 (±0.004)	0.332
290	30	320	12.103 (±0.016)	0.514 (±0.163)	0.417 (±0.009)	0.460	0.333 (±0.631)	0.314 (±0.014)	0.323
320	30	350	14.876 (±0.096)	0.511 (±0.520)	0.409 (±0.018)	0.454	0.329 (±1.430)	0.305 (±0.654)	0.317
350	30	380	18.955 (±0.154)	0.502 (±0.612)	0.405 (±0.122)	0.448	0.321 (±0.996)	0.300 (±0.216)	0.310

Table 12 Sensitivity analysis of the TIE algorithm by keeping $f1$ fixed and varying $f2$ (in the case of HAC)

$f1$	$f2$	Data set	t_{run} $(\pm\sigma)$ $(mins)$	LP (average $\pm\sigma$)	LR (average $\pm\sigma$)	LF	HP (average $\pm\sigma$)	HR (average $\pm\sigma$)	HF
200	30	230	5.024 (±0.093)	0.528 (±0.017)	0.438 (±0.312)	0.479	0.355 (±0.101)	0.326 (±0.088)	0.340
200	60	260	9.986 (±0.021)	0.503 (±0.188)	0.410 (±0.512)	0.452	0.349 (±0.220)	0.317 (±0.096)	0.332
200	90	290	12.645 (±0.003)	0.499 (±1.019)	0.400 (±0.013)	0.444	0.328 (±0.060)	0.314 (±0.054)	0.321
200	120	320	19.832 (±0.005)	0.435 (±0.016)	0.375 (±0.305)	0.403	0.294 (±0.089)	0.285 (±1.001)	0.289
200	150	350	22.608 (±0.014)	0.421 (±0.550)	0.331 (±0.007)	0.371	0.254 (±0.005)	0.263 (±0.043)	0.258
200	180	380	28.111 (±0.119)	0.408 (±0.715)	0.311 (±0.013)	0.353	0.248 (±0.021)	0.247 (±0.167)	0.247

Table 13 Sensitivity analysis of the TIE algorithm by varying $f1$ and keeping $f2$ fixed (in the case of bisect K-means)

$f1$	$f2$	Data set	t_{run} $(\pm\sigma)$ $(mins)$	LP (average $\pm\sigma$)	LR (average $\pm\sigma$)	LF	HP (average $\pm\sigma$)	HR (average $\pm\sigma$)	HF
200	30	230	4.501 (±0.185)	0.377 (±0.003)	0.330 (±0.057)	0.352	0. 257 (±0.002)	0. 219 (±0.076)	0.236
230	30	260	7.001 (±0.009)	0. 365 (±0.018)	0.325 (±0.041)	0.344	0. 250 (±0.013)	0. 213 (±0.041)	0.230
260	30	290	10.965 (±0.014)	0. 360 (±0.154)	0. 311 (±0.190)	0.334	0.236 (±0.009)	0. 202 (±0.199)	0.218
290	30	320	11.909 (±0.006)	0. 359 (±0.201)	0. 304 (±0.301)	0.329	0. 232 (±0.198)	0.200 (±0.031)	0.215
320	30	350	14.132 (±0.014)	0.347 (±0.109)	0. 301 (±0.188)	0.322	0. 219 (±0.005)	0. 194 (±0.005)	0.206
350	30	380	16.502 (±0.176)	0. 342 (±0.005)	0.299 (±0.004)	0.319	0. 206 (±0.203)	0.191 (±0.042)	0.198

Table 14 Sensitivity analysis of the TIE algorithm by keeping $f1$ fixed and varying $f2$ (in the case of bisect K-means)

$f1$	$f2$	Data set	t_{run} $(\pm\sigma)$ $(mins)$	LP (average $\pm\sigma$)	LR (average $\pm\sigma$)	LF	HP (average $\pm\sigma$)	HR (average $\pm\sigma$)	HF
200	30	230	4.501 (±0.185)	0.377 (±0.003)	0.330 (±0.057)	0.352	0.257 (±0.002)	0. 219 (±0.076)	0.236
200	60	260	8.614 (±0.064)	0.362 (±0.011)	0. 316 (±0.054)	0.337	0. 234 (±0.109)	0. 214 (±0.003)	0.224
200	90	290	13.113 (±0.050)	0. 359 (±0.009)	0. 301 (±0.087)	0.327	0. 200 (±0.122)	0. 185 (±0.001)	0.192
200	120	320	17.340 (±0.088)	0. 328 (±0.055)	0. 276 (±0.121)	0.300	0. 174 (±0.034)	0.166 (±0.021)	0.170
200	150	350	21.877 (±0.176)	0. 315 (±0.062)	0. 263 (±0.139)	0.287	0. 155 (±0.142)	0. 154 (±0.046)	0.154
200	180	380	27.190 (±0.316)	0. 303 (±0.006)	0.244 (±0.046)	0.270	0.121 (±0.078)	0.138 (±0.008)	0.129

5.5 Discussion

The observations related to the comparison of taxonomy evolution with taxonomy regeneration presented in Section 5.4 are given below:

- We can see that the time-based evaluation, given in Table 5 in the case of HAC and Table 8 in the case of bisect K-means, clearly reflects the essence of the incremental evolution of taxonomy in comparison to regeneration. Note that as expected, the amount of time it takes to evolve is less than that of regeneration, which is also demonstrated through complexity analysis presented in Section 4.3. This difference in time is more pronounced particularly with the increase in data set size.
- In the case of quality-based evaluation, precision, recall and F-measure values (both lexical, i.e., LP, LR and LF and hierarchical, i.e., HP, HR and HF) for regeneration are slightly better than those obtained as a result of evolution, as shown in Tables 6 and 7 in the case of HAC and Tables 9 and 10 in the case of bisect K-means. Moreover, from the graphs shown in Figs. 8 and 9 in the case of HAC and in Figs. 10 and 11 in the case of bisect K-means, we can observe that the lexical measures are giving better results as compared to the hierarchical measures in all the cases for both regeneration and evolution.
- In the case of quality and time-based evaluation, we can see from all the graphs shown in Figs. 2–7 that the quality-time ratio (i.e., r_{QT}) value is higher in the case of evolution for different size data sets. However, the ratio shows a steady behavior for regeneration which seems not affected much by the increase in the data set size. For evolution case, the r_{QT} is particularly higher when the data set is small and a number of newly added documents is less. As more documents are added to the system, the r_{QT} for evolution shows a decline yet keeping an edge over regeneration.
- Additionally, we can observe that the values for the quality measures in the case of HAC, shown in Tables 6 and 7, appear to be better as compared to that of bisect K-means, shown in Tables 9 and 10 in all the cases for both regeneration and evolution. However, bisect K-means is giving a slight edge over the runtime as compared to that of HAC in all the cases for both regeneration and evolution, as can be seen from Tables 5 and 8.

The observations related to the sensitivity analysis of taxonomy evolution presented in Section 5.4 are given below:

- It can be observed through the results presented in Table 11 in the case of HAC and Table 13 in the case of bisect K-means that when the factor $f1$, i.e., a number of documents used to form the existing taxonomy is varied and the factor $f2$, i.e., a number of newly added documents in the data set is kept fixed, then the increase in time and decrease in taxonomy quality both lexical and hierarchical are showing steady deteriorating behavior with the increasing values of $f1$.

- On the other hand, it can be observed through the results presented in Table 12 in the case of HAC and Table 14 in the case of bisect K-means that when the factor $f1$, i.e., a number of documents used to form the existing taxonomy is kept fixed and the factor $f2$, i.e., a number of newly added documents in the data set is varied, then the increase in time and decrease in taxonomy quality both lexical and hierarchical are showing sharp deteriorating behavior with the increasing value of $f2$.
- Based on this analysis, we can say that a number of newly added documents in the data set (i.e., $f2$) is the factor that has influenced the time and quality metrics values of the TIE algorithm more as compared to a number of documents used to form the existing taxonomy (i.e., $f1$). It seems that the evolution is advantageous when new c large chunk, then there might arise a time wl can lose its advantage over taxonomy regenera

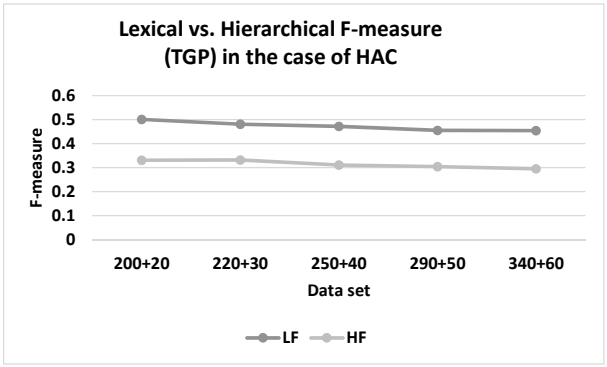


Fig. 8 Graph for Lexical vs. Hierarchical F-measure (TGP) in the case of HAC

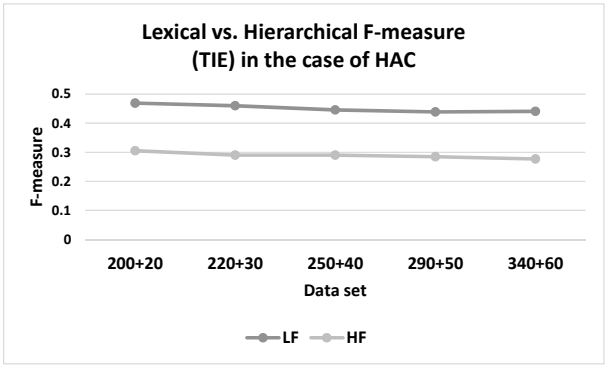


Fig. 9 Graph for Lexical vs. Hierarchical F-measure (TIE) in the case of HAC

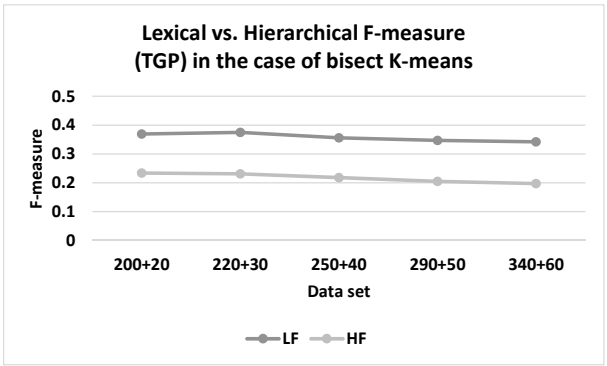


Fig. 10 Graph for Lexical vs. Hierarchical F-measure (TGP) in the case of bisect K-means

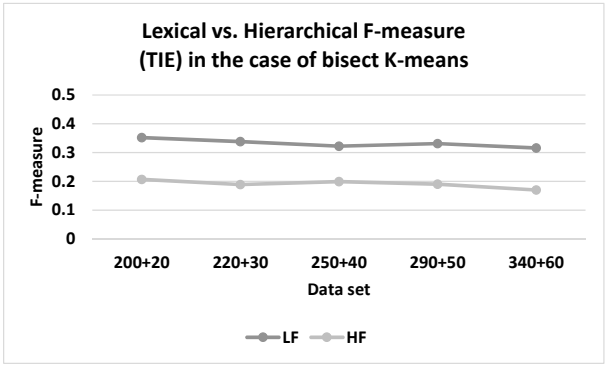


Fig. 11 Graph for Lexical vs. Hierarchical F-measure (TIE) in the case of bisect K-means

The general observations related to the experimental results presented in Section 5.4 are given below:

- The application of the TIE algorithm on existing taxonomies generated through agglomerative and divisive approaches to hierarchical clustering without making any adjustments shows that the

proposed algorithm easily serves as a layer over any of the existing clustering-based taxonomy generation techniques, which is also demonstrated through complexity analysis presented in Section 4.3.

- As far as the scalability of the TIE algorithm is concerned, overall, we can see deteriorating trend in time and quality metrics values with the increasing size of the data set which is used to form taxonomy in all the tests performed, i.e., the test results showing the comparison of taxonomy evolution with regeneration (Tables 5–10) and the test results showing sensitivity analysis of taxonomy evolution (Tables 11–14). This is even the case for taxonomy regeneration from scratch in all the test results shown in Tables 5–10.

6 Conclusion and Future Work

In this paper, taxonomy incremental evolution (TIE) algorithm is designed to incrementally evolve an existing taxonomy to adjust changes that occur in underlying data. The algorithm takes an existing taxonomy, the respective hierarchical structure and newly arrived documents as its input. It identifies the closest cluster in the hierarchy for each of the newly arrived documents based on similarity among them. It then applies various reorganization operators to adjust the newly arrived documents in the hierarchy and finally in the existing taxonomy to evolve it. The algorithm is compared with taxonomy regeneration approach based on complexity analysis and empirical evaluation. The complexity analysis of the algorithm demonstrates that it is better than regeneration in terms of time, and it is independent of underlying clustering approach used for taxonomy generation. On the other hand, the empirical evaluation is performed using a data set of scholarly articles selected from computing domain, based on three parameters. The time-based evaluation clearly shows that the TIE algorithm takes comparatively less time to adjust new documents in an existing taxonomy. Although, taxonomy regeneration is showing better results quality-wise, but the quality-time ratio of the TIE algorithm indicates that the rate of improvement in taxonomy quality per unit time is better than that of regeneration. Moreover, it is identified through the results of the sensitivity analysis of the TIE algorithm that it performs better when the arrival of new data is in small chunk.

The proposed TIE algorithm produces an evolved taxonomy in a shorter period of time to represent changes occur in underlying data for effective utilization of taxonomy. The quality of the evolved taxonomy can be further enhanced by incorporating semantics in identifying the closest cluster for newly arrived documents. Furthermore, the proposed algorithm also requires the setting of time interval value between two consecutive runs of the evolution process. Currently, for the test data set, the running of the TIE algorithm is self-controlled based on the addition of new documents, but for a real data set it should be set depending upon the update characteristics (i.e., a number of newly arrived documents per unit time) of the data set. In general, for the data set for which new documents are frequently arriving the time interval for the evolution should be set lesser than the time interval for the data set for which the arrival of new documents is not very frequent. Moreover, the deteriorating trend in time and quality metrics values with the increasing size of the data set

which is used to form taxonomy shows that the scalability aspect of the proposed solution should also be improved further. In future, the proposed algorithm can be applied to discover emerging trends and patterns in social media where data is evolving rapidly.

References

Baeza-Yates, R., Ribeiro-Neto, B., 2011. *Modern Information Retrieval-the Concepts and Technology behind Search* (Second ed.). Pearson Education Limited.

Blumberg, R., Atre, S., 2003. The problem with unstructured data. *DM REVIEW, 13*, 42-49.

Camina, S. L., 2010. *A comparison of taxonomy generation techniques using bibliometric methods: applied to research strategy formulation.* Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. Massachusetts Institute of Technology.

Carmel, D., Roitman, H., Zwerdling, N., 2009. Enhancing cluster labeling using wikipedia. *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 139-146). Boston, MA, USA.

Cha, S.-H., 2007. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences, 1*(4), 300-307.

Cimiano, P., Hotho, A., Staab, S., 2005. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research, 24*(1), 305-339.

Dawelbait, G., Mezher, T., Woon, W. L., *et al.*, 2010. Taxonomy based trend discovery of renewable energy technologies in desalination and power generation. *Proceedings of the 2010 Technology Management for Global Economic Growth (PICMET)*, (pp. 1-8). Phuket, Thailand.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., *et al.*, 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41*(6), 391-407.

Dietz, E.-A., Vandic, D., Frasincar, F., 2012. TaxoLearn: A semantic approach to domain taxonomy learning. *Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT), 1*, pp. 58-65. Macau, China.

Engel, W., Pryde, C., Sappington, P., 2010. Method and system for enhanced taxonomy generation. *USA Patent No. US 2010/0274733 A1*.

Fountain, T., Lapata, M., 2012. Taxonomy induction using hierarchical random graphs. *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT '12)*, (pp. 466-476). Stroudsburg, PA, USA.

Glover, E., Pennock, D. M., Lawrence, S., *et al.* 2002. Inferring hierarchical descriptions. *Proceedings of the 11th International Conference on Information and Knowledge Management (CKIM)*, (pp. 507-514). McLean, VA, USA.

Hedden, H., 2010. *The Accidental Taxonomist.* Information Today Inc.

Irfan, R., Khan, S., 2016. TIE: An algorithm for incrementally evolving taxonomy for text data. *Proceedings of the 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, (pp. 687-692). Anaheim, California, USA.

Jain, A. K., Murty, M. N., Flynn, P. J., 1999. Data clustering: A review. *ACM Computing Surveys, 31*(3), 264-323.

Karypis, M. S., Kumar, V., Steinbach, M., 2000. A comparison of document clustering techniques. *TextMining Workshop at 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2000)*. Boston, MA, USA.

Kashyap, V., Ramakrishnan, C., Thomas, C., 2005. TaxaMiner: An experimentation framework for automated taxonomy bootstrapping. *International Journal of Web and Grid Services, 1*(2), 240-266.

Koff, W., Gustafson, P., 2012. *Data Revolution.* Tech. rep., Computer Sciences Corporation.

Kumar, A. A., Chandrasekhar, S., 2012. Text data pre-processing and dimensionality reduction techniques for document clustering. *International Journal of Engineering Research and Technology, 1*, pp. 1-6.

Lefever, E., 2015. LT3: A multi-modular approach to automatic taxonomy construction. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, (pp. 944-948). Denver, Colorado, USA.

Li, T., Anand, S. S., 2009. Exploiting domain knowledge by automated taxonomy generation in recommender systems. In *E-Commerce and Web Technologies* (Vol. 5692, pp. 120-131). Springer Berlin Heidelberg.

Manning, C. D., Raghavan, P., Schutze, H., 2008. *Introduction to Information Retrieval.* New York, NY, USA: Cambridge University Press.

Marcacini, R. M., Rezende, S. O., 2010. Incremental construction of topic hierarchies using hierarchical term clustering. *Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE).* Redwood City, San Francisco Bay, CA, USA.

Medelyan, O., Manion, S., Broekstra, J., *et al.*, 2013. Constructing a focused taxonomy from a document collection. In *The Semantic Web: Semantics and Big Data* (Vol. 7882, pp. 367-381). Springer Berlin Heidelberg.

Meijer, K., Frasincar, F., Hogenboom, F., 2014. A semantic approach for extracting domain taxonomies from text. *Decision Support Systems, 62*, 78-93.

Muller, A., Dorre, J., Gerstl, P., *et al.*, 1999. The TaxGen framework: Automating the generation of a taxonomy for a large document collection. *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences (HICSS), Track2*, pp. 9--pp.

Nadkarni, P. M., Ohno-Machado, L., Chapman, W. W., 2011. Natural language processing: An introduction. *Journal of the American Medical Informatics Association, 18*(5), 544-551.

Neshati, M., Alijamaat, A., Abolhassani, H., *et al.*, 2007. Taxonomy learning using compound similarity measure. *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI '07)*, (pp. 487-490). Silicon Valley, USA.

Paukkeri, M.-S., Garcia-Plaza, A. P., Fresno, V., *et al.*, 2012. Learning a taxonomy from a set of text documents. *Applied Soft Computing , 12*(3), 1138-1148.

Qi, X., Yin, D., Xue, Z., Davison, B. D., 2010. Choosing your own adventure: Automatic taxonomy generation to permit many paths. *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM)*, (pp. 1853-1856). Toronto, ON, Canada.

Sanchez, D., Moreno, A., 2004. Automatic generation of taxonomies from the WWW. In *Practical Aspects of Knowledge Management* (Vol. 3336, pp. 208-219). Springer Berlin Heidelberg.

Spangler, W. S., Kreulen, J. T., Newswanger, J. F., 2006. Machines in the conversation: Detecting themes and trends in informal communication streams. *IBM Systems Journal, 45*(4), 785-799.

Sujatha, R., Krishna Rao, B. R. 2011. Taxonomy construction techniques--Issues and challenges. *Indian Journal of Computer Science and Engineering, 2*(5), 661-671.

Thada, V., Jaglan, D. V., 2013. Comparison of jaccard, dice, cosine similarity coefficient to find best fitness value for Web retrieved documents using genetic algorithm. *International Journal of Innovations in Engineering and Technology, 2*(4), 202-205.

Treeratpituk, P., Callan, J. 2006. Automatically labeling hierarchical clusters. *Proceedings of the 2006 International Conference on Digital Government Research*, (pp. 167-176). San Diego, California, USA.

Turner, V., 2014. *The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things*. Tech. rep., EMC Digital Universe with Research and Analysis by International Data Corporation (IDC).

Velardi, P., Sclano, F., 2007. Termextractor: a web application to learn the common terminology of interest groups and research communities. *Proceedings of the 3rd International Conference on Interoperability for Enterprise Software and Applications (I-ESA)*, (pp. 85-94). Funchal, Portugal.

Velardi, P., Faralli, S., Navigli, R., 2013. OntoLearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics, 39*(3), 665-707.

Weng, S.-S., Liu, C.-K., 2004. Using text classification and multiple concepts to answer e-mails. *Expert Systems with Applications, 26*(4), 529-543.

Yang, H.-C., Lee, C.-H., Hsiao, H.-W., 2015. Incorporating self-organizing map with text mining techniques for text hierarchy generation. *Applied Soft Computing, 34*, 251-259.

Yao, J., Cui, B., Cong, G., *et al.*, 2012. Evolutionary taxonomy construction from dynamic tag space. *World Wide Web, 15*(5-6), 581-602.